Algoritmos y Estructuras de Datos II

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

TP 1 : Especificación

KKOS

Integrante	LU	Correo electrónico
Burdman, Kevin	18/22	burdmankevin@gmail.com
Frau, Kenneth	189/22	kenneth.frau@gmail.com
Majic, Santiago	644/22	santiago.majic@protonmail.com
Kerbs, Octavio	64/22	octaviokerbs@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Decisiones Tomadas

Decidimos separar el TAD General en 2 TADs distintos. El primero, LollaPatuza, se encarga de la gestión común a todos los puestos. Sus observadores son todos los Puestos de Comida que están en el festival, las Personas del Festival, los precios base comunes a todos los puestos y todas las compras asociadas a una persona. Las personas que definen como habilitadas en el TP son las personas que estan en el festival, que, en esta especificación, ya viene inicializada con el lollapatooza y no puede ser modificada. Las compras son diferenciadas por item, puesto y cantidad: Si se compran 2 de un mismo item y 1 de otro en un puesto, serán consideradas como 2 compras distintas, con el objetivo de simplificar a la hora de aplicar un descuento.

El TAD PuestoDeComida se encarga de las operaciones más pequeñas referidas al Lollapatooza. Lo identifican su Menu (es decir, los productos que vende), el Stock de cada item que esté en el Menu del Puesto, y los Descuentos que aplican a partir de una compra de una cantidad mínima de un item. Aquí se puede hacer un bypass de los Precios base, porque no restringimos que los descuentos se hagan con 2 o mas items en una misma compra. Si un puesto tuviese un único descuento de 50 % para la compra de más de un item, es esencialmente lo mismo que tener un precio base distinto al de otros puestos que venden tal item.

La principal observación que queremos mencionar refiere al Stock de un item: Este se inicializa con el puesto de comida, y no es nuestro objetivo manejar el stock de un puesto, simplemente registrarlo, por lo que no tenemos, por ejemplo, la operacion de agregar a stock. Lo que si hacemos es restar stock cada vez que se hace una compra del producto. El problema sucede cuando se realiza la operacion de hackear: como esta operación hace que una compra a la que no se aplicó descuento desaparezca del sistema, como si nunca hubiese sucedido. Esto afecta el stock, en el sentido de que se le suman los productos que ya se compraron. Esto genera el problema de que el stock real de un producto puede no ser el mismo que es registrado en el sistema. El caso extremo de esto es si hay 0 de stock real y en el sistema figura como que todavia hay stock. Como el hackeo se considera como un evento imprevisto en el contexto del sistema, y no nos piden como lidiar con sus consecuencias, asumimos que este problema no es de nuestra incumbencia.

2. TAD Lollapatuza

TAD Lollapatuza

igualdad observacional ($\forall$ l, l‘ : lolla) (l $=_{\text{obs}}$ l‘ $\iff$ PuestosDeComida(l) $=_{\text{obs}}$ PuestosDeComida(l‘) $\wedge$ PersonasEnFestival(l) $=_{\text{obs}}$ PersonasEnFestival(l') $\wedge_L$(($\forall$ p : persona) ($p \in$ PersonasEnFestival(l) $\Rightarrow_L$ Compras(l,p) $=_{\text{obs}}$ Compras(l',p))) $\wedge_L$ (($\forall$ i : Item)(($\exists$ pc : PuestoDeComida)(pc $\in$ PuestosDeComida(l) $\wedge_L$ $i \in$ Menu(pc))) $\Rightarrow_L$ (PrecioBase(l,i) $=_{\text{obs}}$ PrecioBase(l‘,i))))

géneros $lolla, lp$

usa Nat, Bool,PuestoDeComida,Precios,multiconj,dicc,conj,Persona,Cantidad,Compra,Item

exporta observadores, generadores, hackeo, ElQueMasPago

observadores básicos

PuestosDeComida : Lollapatuza $\longrightarrow$ conj(PuestoDeComida)

PrecioBase : LollaPatuza lp $\times$ Item i $\longrightarrow$ Precio
$\{(\exists\ p : \text{PuestoDeComida})(p \in \text{PuestosDeComida}(lp) \wedge_L i \in \text{Menu}(p)\}$

PersonasEnFestival : Lollapatuza lp $\longrightarrow$ Personas

Compras : Lollapatuza lp $\times$ Persona pe $\longrightarrow$ multiconj(Compra)
$\{pe \in \text{PersonasEnFestival}(lp)\}$

generadores

LollaPatuza : PuestosDeComida pcs × Personas × Precios pr $\longrightarrow$ Lollapatuza

$$\left\{\begin{array}{l}(\forall\ pc\text{: PuestoDeComida})(pc \in pcs\ \Rightarrow\ _L\ (\forall\ i\text{: Item})(i \in \text{Menu}(pc)\ \Rightarrow\ _L \\ \text{def?}(i,pr)))\end{array}\right\}$$

Comprar : Lollapatuza lp × Persona pe × PuestoDeComida pc × Item i × Cantidad c $\longrightarrow$ Lollapatuza

$$\left\{\begin{array}{l}(pc \in \text{PuestosDeComida}(lp) \wedge pe \in \text{PersonasEnFestival}(lp)) \wedge_L i \in \text{Menu}(pc) \\ \wedge_L \text{obtener}(i,\text{Stock}(pc)) \geq c\end{array}\right\}$$

Hackeo : Lollapatuza lp × Persona per × Item i $\longrightarrow$ Lollapatuza

$$\left\{\begin{array}{l}(\text{pe} \in \text{PersonasEnFestival(lp)} \wedge_L \neg\emptyset?(\text{Compras}(lp,per)) \wedge_L (\exists\ \text{com} : \\ \text{Compra})(com \in \text{Compras}(lp,per) \wedge_L com_1 = i \wedge \text{noTieneDescuento}(lp,com))\end{array}\right\}$$

otras operaciones

ElQueMasPago : Lollapatuza lp $\longrightarrow$ Persona

axiomas

($\forall$ pc : PuestoDeComida, per : Persona, i : Item, c : Cantidad, com : Compra, con : Personas, pr : Precios)

PuestosDeComida(LollaPatuza(pcs,con,pr)) $\equiv$ pcs

PrecioBase(LollaPatuza(pcs,con,pr),i) $\equiv$ obtener(i,pr)

PersonasEnFestival(LollaPatuza(pcs,con,pr)) $\equiv$ con

Compras(LollaPatuza(pcs,con,pr),per) $\equiv$ $\emptyset$

PuestosDeComida(Comprar(lp,per,pc,i,c)) $\equiv$

if dameUno(PuestosDeComida(lp))p = pc
then
Ag(Vender(pc,i,c),sinUno(PuestosDeComida(lp)))
else
Ag(p,sinUno(PuestosDeComida(Comprar(lp,per,pc,i,c))))
end if

PrecioBase(Comprar(lp,per,pc,i,c),i) $\equiv$ PrecioBase(lp,i)

PersonasEnFestival(Comprar(lp,per,pc,i,c)) $\equiv$ PersonasEnFestival(lp)

Compras(Comprar(lp,per,pc,i,c),per') $\equiv$

if $per \neq per'$ **then**
Compras(lp,per')
else
Ag((pc,i,c),Compras(lp,per'))
end if

PesonasEnFestival(hackeo(lp, pe, i)) $\equiv$ PersonasEnFestival(lp)

PuestosDeComida(hackeo(lp, pe, i)) $\equiv$

ReincorporarStock((BuscarCompraSinDescuento(lp,Compras(lp,pe),i)c)$_0$,c_1,1)

BuscarCompraSinDescuento : LollaPalooza lp × multiconj(Compra) $compras$ × Item i $\longrightarrow$ Compra

$$\left\{\begin{array}{l}(\text{pe} \in \text{PersonasEnFestival(lp)} \wedge_L \neg\emptyset?(\text{Compras}(lp,per)) \wedge_L (\exists\ \text{com} : \\ \text{Compra})(com \in \text{Compras}(lp,per) \wedge_L com_1 = i \wedge \text{noTieneDescuento}(lp,com))\end{array}\right\}$$

BuscarCompraSinDescuento $\equiv$

if NoTieneDescuento(dameUno(Compras(lp,pe)))$c \wedge c_1 =$ i **then**
c
else
BuscarCompraSinDescuento(lp,sinUno(Compras(lp,pe),i))
end if

Compras(hackeo(lp,per,i),per') $\equiv$
if $per \neq per'$ **then**
Compras(lp,per')
else
if (BuscarCompraSinDescuento(lp,Compras(lp,pe),i)c)$_2$ $= 1$ **then**
Compras(lp,per') $-\{c\}$
else
Ag(($c_0, c_1, c_2 - 1$),Compras(lp,per')-$\{c\}$)
end if
end if

noTieneDescuento : LollaPatooza lp $\times$ Compra com $\longrightarrow$ Bool
$\{com_0 \in \text{PuestosDeComida}(lp) \wedge_L com_1 \in \text{Menu}(com_0)\}$
noTieneDescuento(lp,com) $\equiv$ AplicarDescuentoItem(com_0,com_1,com_2,
PrecioBase(lp,com_1)) = PrecioBase(lp) $\times$ com_2
ElQueMasPago(lp) $\equiv$ dameUno(LosQueMasPagaron(lp,personasEnFestival(lp)))
LosQueMasPagaron : LollaPatooza lp $\times$ Personas $conj$ $\longrightarrow$ Personas $\{\neg\emptyset(conj)\}$
LosQueMasPagaron(lp,$conj$) $\equiv$
if $\#conj = 1$ **then**
$conj$
else
if sumaTotal(lp,dameUno($conj$)p,Compras(lp,p)) $>$ sumaTotal(dameUno(LosQueMasPagaron(lp,sinUno($conj$))))
then
Ag(p, $\emptyset$)
else
if (sumaTotal(lp,dameUno($conj$)p,Compras(lp,p)) = sumaTotal(dameUno(LosQueMasPagaron(lp,sinUno($conj$))))
then
Ag(p,LosQueMasPagaron(lp,sinUno($conj$)))
else
LosQueMasPagaron(lp,sinUno($conj$))
end if
end if
end if

sumaTotal : LollaPatooza lp $\times$ Persona p $\longrightarrow$ Nat $\{p \in \text{personasEnFestival}(lp)\}$
sumaTotal(lp,per) $\equiv$ SumaTotalAux(lp,per,Compras(lp,per))
sumaTotalAux : LollaPatooza lp $\times$ Persona p $\times$ multiconj(Compra) $compras$ $\longrightarrow$ Nat
$\{compras = \text{Compras}(lp,p)\}$
sumaTotalAux(lp,per,$compras$) $\equiv$
if $\emptyset$?(compras) **then** 0
else AplicarDescuentoItem(((dameUno($Compras$(lp,per))com)$_0$,
com_1,com_2,PrecioBase(lp,com_1))) +
sumaTotalAux(lp,per,sinUno($Compras$(lp,per)))
end if

Fin TAD

TAD Compra es tupla(PuestoDeComida,Item,Cantidad)(indexada en 0)

TADs Item,Persona son String
TAD Personas es conj(Persona)
TADs Cantidad, Precio son Nat
TAD Precios es dicc(Item,Precio)

3. TAD PuestoDeComida

TAD PuestoDeComida

igualdad observacional $PC_1 =_{\text{obs}} PC_2 \iff (\text{Menu}(PC_1) =_{\text{obs}} \text{Menu}(PC_2) \wedge_L (\text{Stock}(PC_1) =_{\text{obs}} \text{Stock}(PC_2) \wedge \text{Descuentos}(PC_1) =_{\text{obs}} \text{Descuentos}(PC_2))$

géneros PC

usa conj, dicc, Menu, Nat, Item, Cantidad, Porcentaje, Precio, Menu,Stock,Descuentos

exporta PC,generadores,observadores,AplicarDescuentoItem

observadores básicos

Menu : PuestoDeComida $\longrightarrow$ conj(Item)
Stock : PuestoDeComida $\longrightarrow$ dicc(Item,Cantidad)
Descuentos : PuestoDeComida p × Item i $\longrightarrow$ dicc(Cantidad,Porcentaje) $\{i \in \text{Menu}(p)\}$

generadores

InaugurarPuesto : Menu m × Stock st × Descuentos ds $\longrightarrow$ PuestoDeComida $\{(\forall\, i : \text{Item})((i \in m \iff \text{def?}(i, st)) \wedge ((\text{def?}(i, st)) \iff \text{def?}(i, ds))\}$
Vender : PuestoDeComida pc × Item i × Cantidad c $\longrightarrow$ PuestoDeComida $\{i \in \text{Menu}(pc) \wedge_L c \leq \text{obtener}(i,\text{Stock}(pc))\}$
ReincorporarStock : PuestoDeComida pc × Item i × Cantidad c $\longrightarrow$ PuestoDeComida $\{i \in \text{Menu}(pc)\}$

otras operaciones

AplicarDescuentoItem : PuestoDeComida pc × Item i × Cantidad c × Precio $\longrightarrow$ natural $\{i \in \text{Menu}(pc) \wedge_L \text{c} \leq \text{obtener}(i,\text{Stock}(pc))\}$

axiomas

($\forall$ pc: PuestoDeComida i : Item, pre : Precio, por : Porcentaje , c,n ; Cantidad)

Menu(InaugurarPuesto(m,st,ds)) $\equiv$ m
Stock(InaugurarPuesto(m,st,ds)) $\equiv$ st
Descuentos(InaugurarPuesto(m,st,ds),i) $\equiv$ obtener(i,ds)
AplicarDescuentoItem($pc.i$,c,pre) $\equiv$

if vacio?(Descuentos(pc,i)) $\vee_L$ cantidadMenorADescuentoMinimo(c,Descuentos(pc,i)) **then**

$pre \times c$

else

AplicarDescuento(c × pre,DescuentoDeMayorCantidad($pc.i$,c))

end if

cantidadMenorADescuentoMinimo : Nat × dicc(Cantidad × Porcentaje) d $\longrightarrow$ bool $\{(\exists\, c : \text{Cantidad})(\text{def?}(c,d))\}$

cantidadMenorADescuentoMinimo(c,*descuentos*) $\equiv$

if $c = 0$ **then**

True

else

if def?(c,*descuentos*) **then**

False

else

cantidadMenorADescuentoMinimo($c-1$,*descuentos*)

end if

end if

DescuentoDeMayorCantidad : PuestoDeComida pc $\times$ Item i $\times$ Cantidad *cantidad* $\longrightarrow$ Nat
$\{(\exists\ n : \text{Nat})(\text{def?}(n,\text{Descuentos}(pc,i)) \wedge_L n \leq \textit{cantidad})\}$

DescuentoDeMayorCantidad(pc,i,c) $\equiv$

if def?(c,Descuentos($pc.i$)) **then**

obtener(c,Descuentos(pc,i))

else

DescuentoDeMayorCantidad(pc,i,$c-1$)

end if

AplicarDescuento : Precio *pre* $\times$ Porcentaje *por* $\longrightarrow$ nat $\{por < 100\}$

AplicarDescuento(*pre*,*por*) $\equiv$ div(*pre*$\times$(100 - *por*),100)

div : Nat n $\times$ Nat k $\longrightarrow$ Nat $\{k > 0\}$

div(n,k) $\equiv$

if $n < k$ **then**

0

else

1 + div($n - k$,k)

end if

Menu(Vender(pc, i, c)) $\equiv$ Menu(pc)

Descuentos(Vender(pc, i, c),i') $\equiv$ Descuentos(pc,i')

Stock(Vender(pc, i, c))) $\equiv$ RemoverCantidadDeStock(Stock(pc),i,c)

RemoverCantidadDeStock : Stock st $\times$ Item i $\times$ Cantidad c $\longrightarrow$ Stock
$\{\text{def?}(i,st) \wedge_L c \leq \text{obtener}(i,st)\}$

RemoverCantidadDeStock(st,i,c) $\equiv$ definir(i,obtener(i,st)$-c$,borrar(i,st))

Menu(ReincorporarStock(pc, i, c)) $\equiv$ Menu(pc)

Stock(ReincorporarStock(pc, i, c))) $\equiv$ AgregarCantidadAStock(Stock(pc),i,c)

AgregarCantidadAStock : Stock st $\times$ Item i $\times$ Cantidad c $\longrightarrow$ Stock $\{\text{def?}(i,st)\}$

AgragarCantidadAStock(st,i,c) $\equiv$ definir(i,obtener(i,st)$+c$,borrar(i,st))

Fin TAD

TAD Menu es conj(Item)
TAD Stock es dicc(Item,Nat)
TAD Descuentos es dicc(Item, dicc(Cantidad, Porcentaje))
TADs Cantidad, Porcentaje son Nat
TAD Item es String